AI CURE PARSEC 4.0

Talasila Dheeraj - 9494710420

Skanda PR - 8792844101

Suhas Raj H R - 8073299734

Problem Statement

To construct an advanced model capable of accurately predicting an individual's heart rate.

Data

The data consists of various psychological features that offer insights into patient's heart rate.

Execution

Step 1: Load the Dataset

```
df= pd.read_csv("train_data.csv")
```

Step 2: Check the Dimensions of Dataset

```
1 df.size
```

```
185000
```

```
1 df.shape
```

```
(5000, 37)
```

Step 3: Determine all the columns of the Dataset

```
1 df.columns
```

```
Index(['uuid', 'VLF', 'VLF_PCT', 'LF', 'LF_PCT', 'LF_NU', 'HF', 'HF_PCT',
       'HF_NU', 'TP', 'LF_HF', 'HF_LF', 'SD1', 'SD2', 'sampen', 'higuci',
       'datasetId', 'condition', 'MEAN_RR', 'MEDIAN_RR', 'SDRR', 'RMSSD',
       'SDSD', 'SDRR_RMSSD', 'HR', 'pNN25', 'pNN50', 'KURT', 'SKEW',
       'MEAN_REL_RR', 'MEDIAN_REL_RR', 'SDRR_REL_RR', 'RMSSD_REL_RR',
       'SDSD_REL_RR', 'SDRR_RMSSD_REL_RR', 'KURT_REL_RR', 'SKEW_REL_RR'],
      dtype='object')
```

Step 4: Remove the columns unnecessary for the Prediction

```
columns_to_remove = ['uuid', 'datasetId']

# Remove the specified columns
df = df.drop(columns=columns_to_remove)
```

Step 5: Check for null values (if any)

```
# Check for null values in the DataFrame
df.isnull().sum()
```

Step 6: Converting the unique states of condition column to numbers.

```
df['condition'].unique()
```

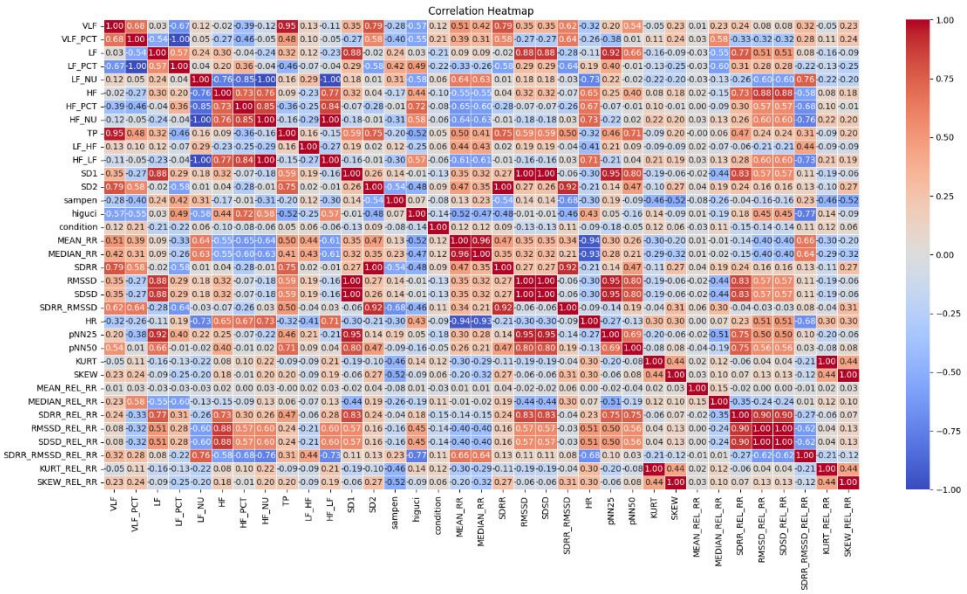
```
array(['interruption', 'no stress', 'time pressure'], dtype=object)
```

```
condition_mapping = {
    'interruption': 1,
    'no stress': 2,
    'time pressure': 3
}

# Replace the values in the 'condition' column with the assigned numbers
df['condition'] = df['condition'].replace(condition_mapping)
```

Step 7: Plot the correlation matrix to see the relation between the columns of the dataset

```
# Calculate the correlation matrix
correlation_matrix = df.corr()
# Create a heatmap using seaborn
plt.figure(figsize=(20, 10))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f", linewidths=.5)
plt.title("Correlation Heatmap")
plt.show()
```



Step 8: Train the Model using 80% of dataset and test it on 20% of the dataset and check for the RMSE (Root Mean Squared Error). The one with lowest RMSE is the best model with high accuracy.

Linear Regression - 1.390535787293412

Ridge Regression - 1.5034607623363505

Lasso Regression - 2.504240125293208

Elastic net Regression - 1.418318992447555

Huber Regression – 1.3974058677930592

RANSAC Regression – 1.795115424053453

Bayesian Regression – 1.3914086302344562

OMP Regression – 3.4393633415258043

XG Boost Regression – 0.4129340732096723

AdaBoost Regression – 1.7008709818816616

Extra Trees– 0.25383346206676904

Decision Tree – 0.7145986744942033

KNN – 2.136906850391142 with 1 neighbour

PCR Regression - 1.4049582852435651

NNN – 0.9341295842084149

Polynomial (deg = 2) – 0.1435064036183413

Polynomial (deg = 3) – 0.06527980802284138

Since the Polynomial Regression of Degree 3 has the lowest RMSE. We have taken it for Final Submission.

run.py Python Script:

In this file, we use the python library 'argparse' to take the path of the test dataset from the user from the command line interface.

The user should use the below command to execute this script:

python run.py --input_file path/to/filename/<test data filename>

When this is called, the test data will be stored as a pandas data frame. Further, the polynomial regression model is generated, using the train dataset, which is further used for predicting the heart rate.

This heart rate data frame generated by the model is stored as 'results.csv' file in the same working directory as the run.py file.

```
parser = argparse.ArgumentParser()
parser.add_argument("--input_file", type=str, default="sample_test_data.csv")
args = parser.parse_args()
```

The above code snippet shows the working of the ‘argparse’ python library.

```
df= pd.read_csv("train_data.csv")

columns_to_remove = ['uuid', 'datasetId']

condition_mapping = {
    'interruption': 1,
    'no stress': 2,
    'time pressure': 3
}

df = df.drop(columns=columns_to_remove)

df['condition'] = df['condition'].replace(condition_mapping)

test_data = pd.read_csv(args.input_file)

test_data['condition'] = test_data['condition'].replace(condition_mapping)

features = ['VLF', 'VLF_PCT', 'LF', 'LF_PCT', 'LF_NU', 'HF', 'HF_PCT', 'HF_NU',
       'TP', 'LF_HF', 'HF_LF', 'SD1', 'SD2', 'sampen', 'higuci', 'condition',
       'MEAN_RR', 'MEDIAN_RR', 'SDRR', 'RMSSD', 'SDSD', 'SDRR_RMSSD',
       'pNN25', 'pNN50', 'KURT', 'SKEW', 'MEAN_REL_RR', 'MEDIAN_REL_RR',
       'SDRR_REL_RR', 'RMSSD_REL_RR', 'SDSD_REL_RR', 'SDRR_RMSSD_REL_RR',
       'KURT_REL_RR', 'SKEW_REL_RR']

target = 'HR'


X_train, X_test, y_train, y_test = train_test_split(df[features], df[target], test_size=0.2, random_state=42)


degree = 3
poly = PolynomialFeatures(degree=degree)
X_train_poly = poly.fit_transform(X_train)
X_test_poly = poly.transform(test_data[features])


model = LinearRegression()


model.fit(X_train_poly, y_train)


predictions = model.predict(X_test_poly)

results_df = pd.DataFrame({'uuid': test_data['uuid'], 'HR': predictions})

results_df.to_csv("results.csv", index=False)
```

This shows the implementation of the prediction model, and the generation of the output.